# Intel® Celeron® M Processor

## Specification Update

*April 2006*

Document Number: 300303-019

# Contents



§

# Revision History

| Revision Number | Description | Date |
|---|---|---|
| -001 | Initial Release | January 2004 |
| -002 | Revisions include:<br>• Added errata WW22- W23<br>• Added Specification Clarification W1<br>• Updated Processor Identification table | April 2004 |
| -003 | Revisions include:<br>• Added errata WW24- W25 | May 2004 |
| -004 | Revisions include:<br>• Updated Processor Identification table | June 2004 |
| -005 | Revisions include:<br>• Change to Title to reflect processors (adding Celeron M processor ULV on 90 nm processor)<br>• Processor Identification table to include Celeron processor ULV on 90 nm process<br>• Added errata W26 and W27 | July 2004 |
| -006 | Revisions include:<br>• Updated General information (see Figure 3) for Celeron processor 350 and 360<br>• Processor Identification table to include Celeron processor 350 and 360 | August 2004 |
| -007 | Revisions include:<br>• Added errata W28 through W33 | November 2004 |
| -008 | Revisions include:<br>• Added errata W34 and W35 | December 2004 |
| -009 | • Updated Processor Identification Table: Added C-0 S-Specs<br>• Updated Summary Tables of Changes<br>• Added Erratum W36 – W38 | February 2005 |
| -010 | • Updated Processor Identification Table | March 2005 |

| Revision Number | Description | Date |
|---|---|---|
| -011 | • Updated Summary Tables of Changes<br>• Updated Processor Identification Table:<br>o Added Celeron M processor ULV 383<br>o Updated Celeron M processor 360J & 350J<br>• Added Errata W39<br>• Added Specification Clarification W1<br>• Added Specification Clarification W2 | May 2005 |
| -012 | • Updated Summary Tables of Changes<br>• Removed Erratum W28 – W30 (which were duplicates of W3 – W5)<br>• Added Erratum W40 – W42 | June 2005 |
| -013 | • Updated Processor Identification Table:<br>o Added Celeron M processor 380 | July 2005 |
| -014 | • Updated Affected and Related Documents Tables<br>• Updated Processor Identification Table 1 | July 2005 |
| -015 | • Updated Processor Identification Table 1 | October 2005 |
| -016 | • Added Erratum W43 | November 2005 |
| -017 | • Added Errata W44 and W45 | December 2005 |
| -018 | • Added Errata W46 – W51<br>• Removed Specification Clarification W1; refer to Section 18.8 of the *IA-32 Intel® Architecture Software Developer's Manual, Volume 3B* for detailed Time Stamp Counter information<br>• Updated Processor Identification Table 1 | January 2006 |
| -019 | • Added Errata W52 – W66<br>• Updated Errata W18, W34<br>• Added General information on Intel Celeron M Processor on 65nm process<br>• Updated Summary of Changes Table to include CPU Signature = 06E8h (Intel Celeron M Processor on 65 nm process)<br>• Updated Processor Identification Table 1<br>• Updated Affected Documents Table | April 2006 |

§

# Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents/Related Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

## Affected Documents

| Document Title | Document Number |
|---|---|
| *Intel® Celeron® M Processor Datasheet* | 300302-003 |
| *Intel® Celeron® M Processor on 65 nm Process Datasheet* | 312726-001 |
| *Intel® Celeron® M Processor on 90 nm Process Datasheet* | 303110-006 |

## Related Documents

| Document Title | Document Number |
|---|---|
| *IA-32 Intel® Architecture Software Developer's Manual, Volume 1: Basic Architecture* | 253665 |
| *IA-32 Intel® Architecture Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M* | 253666 |
| *IA-32 Intel® Architecture Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z* | 253667 |
| *IA-32 Intel® Architecture Software Developer's Manual, Volume 3A: System Programming Guide* | 253668 |
| *IA-32 Intel® Architecture Software Developer's Manual, Volume 3B: System Programming Guide* | 253669 |

## Nomenclature

**S-Spec Number** is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc. as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

**Errata** are design defects or errors. Errata may cause the Intel® Celeron® M processor's behavior to deviate from published specifications. Hardware and software, designed to be used with any given processor, must assume that all errata documented for that processor are present on all devices unless otherwise noted.

**Documentation Changes** include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

**Specification Clarifications** describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

**Specification Changes** are modifications to the current published specifications for the Celeron M processor. These changes will be incorporated in the next release of the specifications.

# Summary Tables of Changes

The following table indicates the errata, documentation changes, specification clarifications, or specification changes that apply to the Celeron M processor. Intel intends to fix some of the errata in a future stepping of the component and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

## Codes Used in Summary Table

### 1.1.1 Stepping

| | |
|---|---|
| X: | Erratum, Specification Change or Clarification that applies to this stepping. |
| (No mark) or (Blank Box): | This erratum is fixed in listed stepping or specification change does not apply to listed stepping. |

### 1.1.2 Status

| | |
|---|---|
| Doc: | Document change or update that will be implemented. |
| PlanFix: | This erratum may be fixed in a future of the product. |
| Fixed: | This erratum has been previously fixed. |
| NoFix: | There are no plans to fix this erratum. |

### 1.1.3 Row

| | |
|---|---|
| Shaded: | This item is either new or modified from the previous version of the document. |

Each specification update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor specification updates:

A = Intel® Pentium® II processor
B = Mobile Intel® Pentium® II processor
C = Intel® Celeron® processor
D = Intel® Pentium® II Xeon® processor
E = Intel® Pentium® III processor
G = Intel® Pentium® III Xeon® processor
F = Intel® Pentium® processor Extreme Edition and Intel® Pentium® D processor

H = Mobile Intel® Celeron® processor at 466 MHz, 433 MHz, 400 MHz, 366 MHz, 333 MHz, 300 MHz, and 266 MHz
J = 64-bit Intel® Xeon® processor MP with 1-MB L2 cache
K = Mobile Intel® Pentium® III Processor – M
L = Intel® Celeron® D processor
M =Mobile Intel® Celeron® processor
N = Intel® Pentium® 4 processor
O = Intel® Xeon® processor MP
P = Intel® Xeon® processor
Q = Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology† on 90-nm technology process
R = Intel® Pentium® 4 processor on 90 nm process
S = 64-bit Intel® Xeon® processor with 800 MHz system bus (1-MB and 2- MB L2 cache versions)
T = Mobile Intel® Pentium® 4 processor – M
U = 64-bit Intel® Xeon® processor MP with up to 8-MB L3 cache
V = Mobile Intel® Celeron® processor on .13 Micron process in micro-FCPGA package
W = Intel® Celeron® M processor
X = Intel® Pentium® M processor on 90 nm process with 2-MB L2 cache
Y = Intel® Pentium® M processor
Z = Mobile Intel® Pentium® 4 processor with 533 MHz system bus

***Note:*** The specification updates for the Pentium® processor, Pentium® Pro processor, and other Intel products do not use this convention.

| NO. | Steppings | | | | Plans | ERRATA |
|---|---|---|---|---|---|---|
| | B-1 CPU Signature = 06D6h | C-0 CPU Signature = 06D8h | B-1 CPU Signature = 0695h | C-0 CPU Signature = 06E8h | | |
| W1 | | | X | | NoFix | Performance Monitoring Event that counts the number of instructions decoded (D0h) is not accurate |
| W2 | | | X | | NoFix | RDTSC Instruction May Report the Wrong Time Stamp Counter Value |
| W3 | | | X | X | NoFix | Code Segment limit violation may occur on 4 Gbyte limit check |
| W4 | | | X | X | NoFix | FST Instruction with Numeric and Null Segment Exceptions may cause General Protection Faults to be Missed and FP Linear Address (FLA) Mismatch |
| W5 | X | X | X | | NoFix | Code Segment (CS) is wrong on SMM Handler when SMBASE is not aligned |
| W6 | X | X | X | X | NoFix | IFU/BSU Deadlock May Cause System Hang |
| W7 | | | X | | NoFix | Processor can enter a livelock condition under certain conditions when FP exception is pending. |

| NO. | Steppings | | | | Plans | ERRATA |
|---|---|---|---|---|---|---|
| | B-1 CPU Signature = 06D6h | C-0 CPU Signature = 06D8h | B-1 CPU Signature = 0695h | C-0 CPU Signature = 06E8h | | |
| W8 | | | X | | NoFix | Write Cycle of Write Combining Memory Type does not Self Snoop |
| W9 | | | X | | NoFix | Performance Monitoring Event that counts Floating Point Computational Exceptions (11h) is not accurate. |
| W10 | | | X | | NoFix | Inconsistent Reporting of Data Breakpoints on FP (MMX) loads |
| W11 | | | X | | NoFix | Code Breakpoint may be taken after POP SS instruction if it is followed by an instruction that faults |
| W12 | | | X | | NoFix | SysEnter and SysExit instructions may write incorrect Requestor Privilege Level (RPL) in the FP Code Segment selector (FCS) |
| W13 | X | X | X | X | NoFix | Memory Aliasing with Inconsistent A and D Bits may Cause Processor Deadlock |
| W14 | X | X | X | | NoFix | RDMSR or WRMSR to Invalid MSR Address May Not Cause GP Fault |
| W15 | | | X | | NoFix | FP Tag Word Corruption |
| W16 | X | X | X | | NoFix | Unable to Disable Reads/Writes to Performance Monitoring Related MSRs |
| W17 | X | X | X | | NoFix | Move to Control Register Instruction May Generate a Breakpoint Report |
| W18 | | | X | | NoFix | REP MOVS Operation in Fast String Mode Continues in That Mode When Crossing Into a Page With a Different Memory Type |
| W19 | | | X | | NoFix | The FXSAVE, STOS, or MOVS Instruction May Cause a Store Ordering Violation When Data Crosses a Page With a UC Memory Type |
| W20 | | | X | | NoFix | Machine Check Exception May Occur Due to Improper Line Eviction in the IFU |
| W21 | | | X | X | NoFix | POPF and POPFD Instructions That Set the Trap Flag Bit May Cause Unpredictable Processor Behavior |
| W22 | | | X | | NoFix | Performance Event Counter Returns Incorrect Value on L2_LINES_IN Event |
| W23 | | | X | X | NoFix | VM Bit Will Be Cleared on Double Fault Handler |

| NO. | Steppings | | | | Plans | ERRATA |
|---|---|---|---|---|---|---|
| | B-1 CPU Signature = 06D6h | C-0 CPU Signature = 06D8h | B-1 CPU Signature = 0695h | C-0 CPU Signature = 06E8h | | |
| W24 | X | X | X | | NoFix | Code Fetch Matching Disabled Debug Register May Cause Debug Exception |
| W25 | X | X | X | | NoFix | Upper Four PAT Entries Not Usable With Mode B or Mode C Paging |
| W26 | X | X | X | | NoFix | SSE/SSE2 Streaming Store Resulting in a Self-Modifying Code (SMC) Event May Cause Unexpected Behavior |
| W27 | X | X | | | NoFix | Error in Instruction Fetch Unit (IFU) Can Result in an Erroneous Machine Check-Exception (#MC) |
| W28 | | | | | | Removed, see Erratum W3 |
| W29 | | | | | | Removed, see Erratum W4 |
| W30 | | | | | | Removed, see Erratum W5 |
| W31 | X | X | X | X | NoFix | Page with PAT (Page Attribute Table) Set to USWC (Uncacheable Speculative Write Combine) While Associated MTRR (Memory Type Range Register) is UC (Uncacheable) May Consolidate to UC |
| W32 | X | X | X | | NoFix | Under Certain Conditions LTR (Load Task Register) Instruction May Result in System Hang |
| W33 | X | X | X | | NoFix | Loading from Memory Type USWC (Uncacheable Speculative Write Combine) May Get Its Data Internally Forwarded From a Previous Pending Store |
| W34 | X | X | X | X | NoFix | FPU Operand Pointer may not be cleared following FINIT/FNINIT |
| W35 | X | X | X | | NoFix | FSTP (Floating Point Store) Instruction Under Certain Conditions May Result In Erroneously Setting a Valid Bit on an FP (Floating Point) Stack Register |
| W36 | | X | | | NoFix | An Execute Disable Bit Violation May Occur on a Data Page-Fault |
| W37 | | X | | | NoFix | CPUID Leaf 0x80000006 May Provide the Incorrect Value for an 8-Way Associative Cache |
| W38 | X | | X | | PlanFix | Snoops during the Execution of a HLT (Halt) Instruction May Lead to Unexpected System Behavior |

| NO. | Steppings | | | | Plans | ERRATA |
|---|---|---|---|---|---|---|
| | B-1 CPU Signature = 06D6h | C-0 CPU Signature = 06D8h | B-1 CPU Signature = 0695h | C-0 CPU Signature = 06E8h | | |
| W39 | X | X | X | X | NoFix | Invalid Entries in Page-Directory-Pointer-Table Register (PDPTR) May Cause General Protection (#GP) Exception if the Reserved Bits are Set to One |
| W40 | X | X | X | X | NoFix | INIT Does Not Clear Global Entries in the TLB |
| W41 | X | X | X | X | NoFix | Use of Memory Aliasing with Inconsistent Memory Type May Cause System Hang |
| W42 | X | X | X | X | NoFix | Machine Check Exception May Occur When Interleaving Code Between Different Memory Types |
| W43 | X | X | X | | NoFix | Split I/O Writes Adjacent to Retry of APIC End of Interrupt (EOI) Request May Cause Livelock Condition |
| W44 | X | X | X | X | No Fix | General Protection (#GP) Fault May Not Be Signaled On Data Segment Limit Violation Above 4G Limit |
| W45 | X | X | X | X | No Fix | DR3 Address Match on MOVD/MOVQ/MOVNTQ Memory Store Instruction May Incorrectly Increment Performance Monitoring Count for Saturating SIMD Instructions Executed (Event B1h) |
| W46 | X | X | | X | No Fix | Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts |
| W47 | X | X | X | | No Fix | Processor INIT# Will Cause a System Hang if Triggered During an NMI Interrupt Routine Performed During Shutdown |
| W48 | X | X | X | X | No Fix | Certain Performance Monitoring Counters Related to Bus, L2 Cache and Power Management are Inaccurate |
| W49 | X | X | X | X | No Fix | CS Limit Violation on RSM May be Serviced before Higher Priority Interrupts/Exceptions |
| W50 | X | X | X | | No Fix | A Write to an APIC Register Sometimes May Appear to Have Not Occurred |
| W51 | X | X | X | | No Fix | The Processor May Report a #TS Instead of a #GP Fault |
| W52 | X | X | | | No Fix | Last Exception Record (LER) MSRs May be Incorrectly Updated |
| W53 | | | | X | No Fix | FP Inexact-Result Exception Flag May Not Be Set |

| NO. | Steppings | | | | Plans | ERRATA |
|---|---|---|---|---|---|---|
| | B-1 CPU Signature = 06D6h | C-0 CPU Signature = 06D8h | B-1 CPU Signature = 0695h | C-0 CPU Signature = 06E8h | | |
| W54 | | | | X | No Fix | MOV With Debug Register Causes Debug Exception |
| W55 | | | | X | Plan Fix | Processor Digital Thermal Sensor (DTS) Readout Stops Updating upon Returning from C3 State |
| W56 | | | | X | No Fix | LOCK# Asserted During a Special Cycle Shutdown Transaction May Unexpectedly De-assert |
| W57 | | | | X | No Fix | Last Branch Records (LBR) Updates May be Incorrect after a Task Switch |
| W58 | | | | X | No Fix | Address Reported by Machine-Check Architecture (MCA) on Single-bit L2 ECC Errors May be Incorrect |
| W59 | | | | X | No Fix | Disabling of Single-step On Branch Operation May be Delayed following a POPFD Instruction |
| W60 | | | | X | No Fix | Performance Monitoring Counters That Count External Bus Events May Report Incorrect Values after Processor Power State Transitions |
| W61 | | | | X | No Fix | VERW/VERR/LSL/LAR Instructions May Unexpectedly Update the Last Exception Record (LER) MSR |
| W62 | | | | X | No Fix | Performance Monitoring Events for Retired Floating Point Operations (C1h) May Not Be Accurate |
| W63 | X | X | | X | No Fix | Global Pages in the Data Translation Look-Aside Buffer (DTLB) May Not Be Flushed by RSM instruction before Restoring the Architectural State from SMRAM |
| W64 | | | | X | No Fix | Data Breakpoint/Single Step on MOV SS/POP SS May Be Lost after Entry into SMM |
| W65 | | | | X | Plan Fix | Hardware Prefetch Performance Monitoring Events May Be Counted Inaccurately |
| W66 | | | | X | NoFix | REP MOVS Operation in Fast String Mode Continues in That Mode When Crossing Into a Page With a Different Memory Type |

| Number | SPECIFICATION CHANGES |
|---|---|
| | There are no Specification Changes in this Specification Update revision. |

| Number | SPECIFICATION CLARIFICATIONS |
|---|---|
| W1 | Specification Clarification with Respect to Time-Stamp Counter - Removed |
| W2 | Thermal Diode Offset Specification Clarification - Removed |

| Number | DOCUMENTATION CHANGES |
|---|---|
| | There are no Documentation Changes in this Specification Update revision. |

§

# *Identification Information*

The Celeron M processor can be identified by the following values:

| Family[1] | Model[2] | Model[2] (Celeron M on 90 nm process) | Brand ID [3] |
|---|---|---|---|
| 0110 | 1001 | 1101 | 00010010 |

**NOTES:**
1. The Family corresponds to bits [11:8] of the EDX register after Reset, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register.
2. The Model corresponds to bits [7:4] of the EDX register after Reset, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register.
3. The Brand ID corresponds to bits [7:0] of the EBX register after the CPUID instruction is executed with a 1 in the EAX register.

**Table 1. Intel Celeron M Identification**

| S-Spec | Processor number | Product Stepping | L2 Cache Size (bytes) | CPU Signature | Core Frequency | Bus Frequency | Voltage | Package<br>Micro-FCBGA-Pb= Micro-FCBGA Lead Free |
|---|---|---|---|---|---|---|---|---|
| SL92F | 430 | C-0 | 1 M | 06E8h | 1.73 | 533 MHz | 1.30-1.00 V | Micro-FCPGA |
| SL8VZ | 420 | C-0 | 1 M | 06E8h | 1.60 | 533 MHz | 1.30-1.00 V | Micro-FCPGA |
| SL9KV | 430 | C-0 | 1 M | 06E8h | 1.73 | 533 MHz | 1.30-1.00 V | Micro-FCBGA |
| SL8XW | 423 | C-0 | 1 M | 06E8h | 1.06 | 533 MHz | 1.10-0.85 V | Micro-FCBGA |
| SL8MH | 390 | C-0 | 1 M | 06D8h | 1.7 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCBGA |
| SL8MP | 390 | C-0 | 1 M | 06D8h | 1.7 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCBGA |
| SL8MV | 390 | C-0 | 1 M | 06D8h | 1.7 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCBGA-Pb |
| SL8LP | 383 | C-0 | 1 M | 06D8h | 1.00 GHz | 400 MHz | 0.876 V – 0.956 V[1] | Micro-FCBGA |
| SL8LV | 383 | C-0 | 1 M | 06D8h | 1.00 GHz | 400 MHz | 0.876 V – 0.956 V[1] | Micro-FCBGA-Pb |
| SL8BP | 383 | C-0 | 1 M | 06D8h | 1.00 GHz | 400 MHz | 0.940 V | Micro-FCBGA |
| SL8BN | 383 | C-0 | 1 M | 06D8h | 1.00 GHz | 400 MHz | 0.940 V | Micro-FCBGA-Pb |
| SL8LQ | 373 | C-0 | 512K | 06D8h | 1.00 GHz | 400 MHz | 0.876 V – 0.956 V[1] | Micro-FCBGA |
| SL8LW | 373 | C-0 | 512K | 06D8h | 1.00 GHz | 400 MHz | 0.876 V – 0.956 V[1] | Micro-FCBGA-Pb |
| SL8A4 | 373 | C-0 | 512K | 06D8h | 1.00 GHz | 400 MHz | 0.940 V | Micro-FCBGA |
| SL89S | 373 | C-0 | 512K | 06D8h | 1.00 GHz | 400 MHz | 0.940 V | Micro-FCBGA-Pb |
| SL8MN | 380 | C-0 | 1 M | 06D8h | 1.60 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCPGA |
| SL8MG | 380 | C-0 | 1 M | 06D8h | 1.60 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCBGA |

| S-Spec | Processor number | Product Stepping | L2 Cache Size (bytes) | CPU Signature | Core Frequency | Bus Frequency | Voltage | Package<br><br>Micro-FCBGA-Pb= Micro-FCBGA Lead Free |
|---|---|---|---|---|---|---|---|---|
| SL8MU | 380 | C-0 | 1 M | 06D8h | 1.60 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCBGA-Pb |
| SL8MM | 370 | C-0 | 1 M | 06D8h | 1.50 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCPGA |
| SL8MF | 370 | C-0 | 1 M | 06D8h | 1.50 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCBGA |
| SL8MT | 370 | C-0 | 1 M | 06D8h | 1.50 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCBGA-Pb |
| SL86J | 370 | C-0 | 1 M | 06D8h | 1.50 GHz | 400 MHz | 1.260 V | Micro-FCPGA |
| SL86P | 370 | C-0 | 1 M | 06D8h | 1.50 GHz | 400 MHz | 1.260 V | Micro-FCBGA |
| SL86D | 370 | C-0 | 1 M | 06D8h | 1.50 GHz | 400 MHz | 1.260 V | Micro-FCBGA-Pb |
| SL8ML | 360J | C-0 | 1 M | 06D8h | 1.40 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCPGA |
| SL8ME | 360J | C-0 | 1 M | 06D8h | 1.40 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCBGA |
| SL86K | 360J | C-0 | 1 M | 06D8h | 1.40 GHz | 400 MHz | 1.260 V | Micro-FCPGA |
| SL86Q | 360J | C-0 | 1 M | 06D8h | 1.40 GHz | 400 MHz | 1.260 V | Micro-FCBGA |
| SL8MK | 350J | C-0 | 1 M | 06D8h | 1.30 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCPGA |
| SL8MD | 350J | C-0 | 1 M | 06D8h | 1.30 GHz | 400 MHz | 1.004 V – 1.292 V[1] | Micro-FCBGA |
| SL86L | 350J | C-0 | 1 M | 06D8h | 1.30 GHz | 400 MHz | 1.260 V | Micro-FCPGA |
| SL86R | 350J | C-0 | 1 M | 06D8h | 1.30 GHz | 400 MHz | 1.260 V | Micro-FCBGA |
| SL7LS | 360 | B-1 | 1 M | 06D6h | 1.4 GHz | 400 MHz | 1.260 V | Micro-FCPGA |
| SL7LR | 360 | B-1 | 1 M | 06D6h | 1.4 GHz | 400 MHz | 1.260 V | Micro-FCBGA |
| SL7RA | 350 | B-1 | 1 M | 06D6h | 1.3 GHz | 400 MHz | 1.260 V | Micro-FCPGA |
| SL7R9 | 350 | B-1 | 1 M | 06D6h | 1.3 GHz | 400 MHz | 1.260 V | Micro-FCBGA |
| SL6N7 | 320 | B-1 | 512K | 0695h | 1.30 GHz | 400 MHz | 1.356 V | Micro-FCPGA |
| SL6NM | 320 | B-1 | 512K | 0695h | 1.30 GHz | 400 MHz | 1.356 V | Micro-FCBGA |
| SL6N6 | 330 | B-1 | 512K | 0695h | 1.40 GHz | 400 MHz | 1.356 V | Micro-FCPGA |
| SL6NL | 330 | B-1 | 512K | 0695h | 1.40 GHz | 400 MHz | 1.356 V | Micro-FCBGA |
| SL7MT | 340 | B-1 | 512K | 0695h | 1.50 GHz | 400 MHz | 1.356 V | Micro-FCBGA |
| SL7ME | 340 | B-1 | 512K | 0695h | 1.50 GHz | 400 MHz | 1.356 V | Micro-FCPGA |
| SL79S | n/a | B-1 | 512K | 0695h | 1.20 GHz | 400 MHz | 1.356 V | Micro-FCPGA |
| SL79T | n/a | B-1 | 512K | 0695h | 1.20 GHz | 400 MHz | 1.356 V | Micro-FCBGA |
| SL7GE | n/a | B-1 | 512K | 0695h | 600 MHz | 400 MHz | 1.004 V | Micro-FCBGA |
| SL7DB | n/a | B-1 | 512K | 0695h | 800 MHz | 400 MHz | 1.004 V | Micro-FCBGA |
| SL7DH | n/a | B-1 | 512K | 0695h | 900 MHz | 400 MHz | 1.004 V | Micro-FCBGA |
| SL7F7 | 353 | B-1 | 512 K | 06D6h | 900 MHz | 400 MHz | 0.940 V | Micro-FCBGA |
| SL7QX | 353 | B-1 | 512K | 06D6h | 900 MHz | 400 MHz | 0.940 V | Micro-FCBGA-Pb |

**NOTES:**

1. These are optimized Voltage ID (VID) values. Individual processor VID values may be calibrated during manufacturing such that two devices at the same speed may have different VID settings.

# Component Marking Information

### Figure 1. The Intel Celeron M Processor (Micro-FCPGA/FCBGA) S-Spec Markings

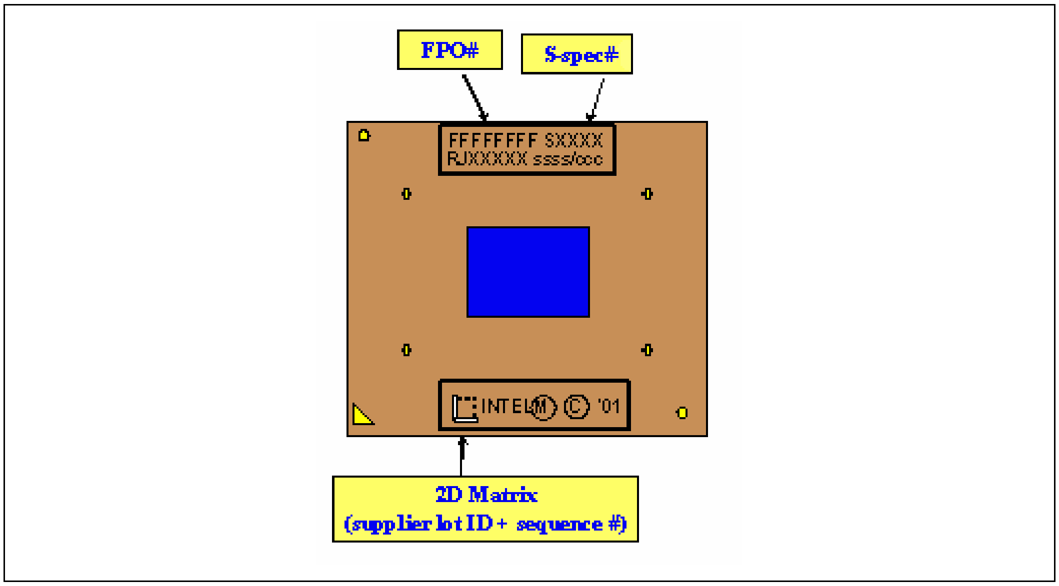


**Figure 2. The Intel Celeron Processor ULV on 90 nm Process (Micro-FCBGA) S-Spec Markings**

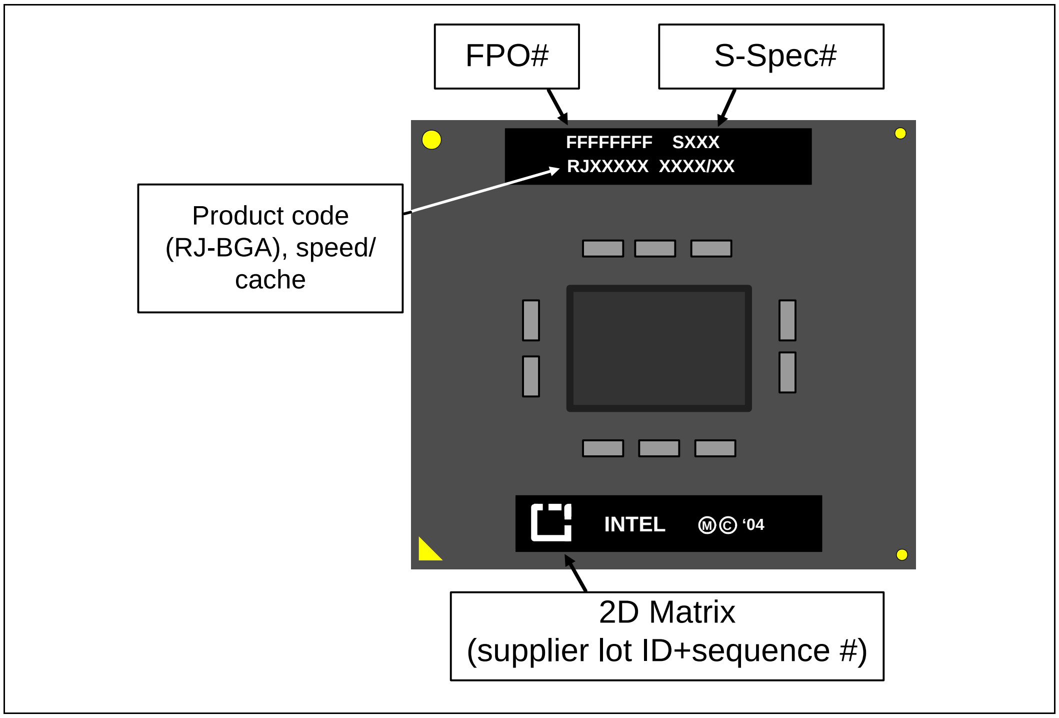
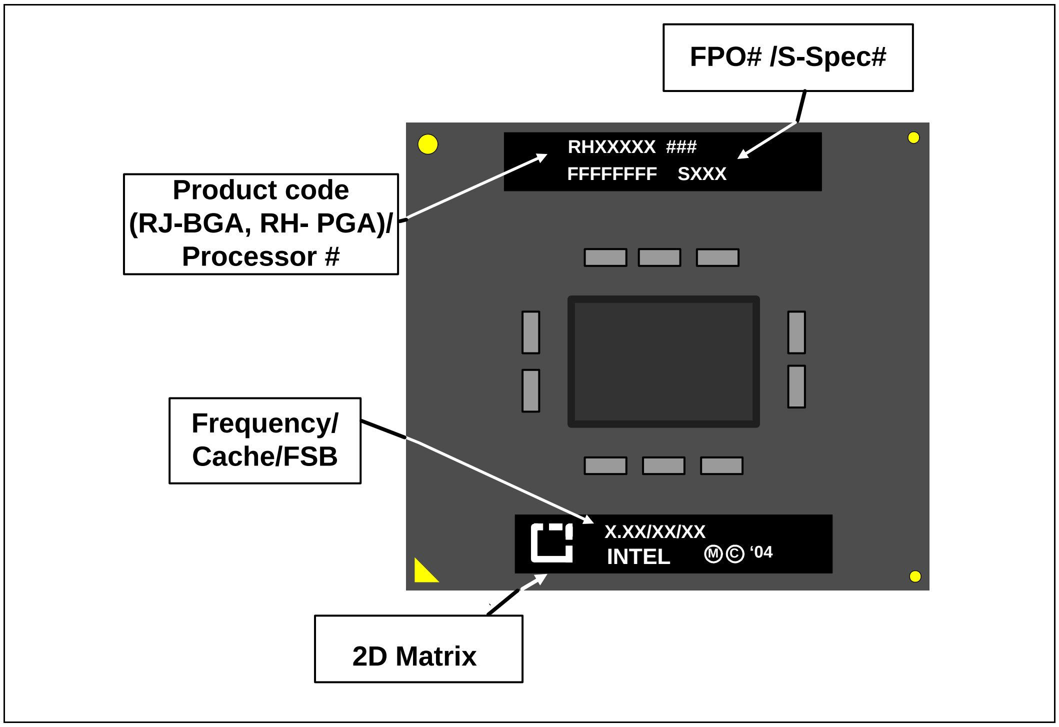


**Figure 3.The Intel Celeron M Processor on 90 nm Process (Micro-FCPGA / FCBGA) S-Spec Markings**




§

# Errata

## W1. Performance Monitoring Event that Counts the Number of Instructions Decoded (D0h) Is Not Accurate

**Problem:** The performance-monitoring event that counts the number of instructions decoded may have inaccurate results.

**Implication:** There is no functional impact of this erratum. However, the results/counts from this performance monitoring event should not be considered as being accurate

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W2. RDTSC Instruction May Report the Wrong Time-Stamp Counter Value

**Problem:** The time-stamp counter is a 64-bit counter that is read in two 32-bit chunks. The counter incorrectly advances and therefore the two chunks may go out of synchronization causing the Read Time-stamp Counter (RDTSC) instruction to report the wrong time-stamp counter value

**Implication:** This erratum may cause software to see the wrong representation of processor time and may result in unpredictable software operation.

**Workaround:** It is possible for BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W3. Code Segment Limit Violation May Occur on 4-Gbyte Limit Check

**Problem:** Code Segment limit violation may occur on 4-Gbyte limit check when the code stream wraps around in a way that one instruction ends at the last byte of the segment and the next instruction begins at 0x0.

**Implication:** This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

**Workaround:** Avoid code that wraps around segment limit.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W4. FST Instruction with Numeric and Null Segment Exceptions May Cause General Protection Faults to Be Missed and FP Linear Address (FLA) Mismatch

**Problem:** FST instruction combined with numeric and null segment exceptions may cause General Protection Faults to be missed and FP Linear Address (FLA) mismatch.

**Implication:** This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes

### W5. Code Segment (CS) Is Wrong on SMM Handler When SMBASE Is Not Aligned

**Problem:** With SMBASE being relocated to a non-aligned address, during SMM entry the CS can be improperly updated which can lead to an incorrect SMM handler.

**Implication:** This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

**Workaround:** Align SMBASE to 32-kB.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W6. IFU/BSU Deadlock May Cause System Hang

**Problem:** A lockable instruction with memory operand that spans across two pages may, given some rare internal conditions, hang the system.

**Implication:** When this erratum occurs, the system may hang. Intel has not observed this erratum with any commercially available software or system.

**Workaround:** Lockable data should always be contained in a single page.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W7. Processor Can Enter a Livelock Condition under Certain Conditions When FP Exception Is Pending

**Problem:** Processor clock modulation may be controlled via a processor register (IA32_THERM_CONTROL) or via the STPCLK# signal. While the processor clock is constantly being actively modulated at 12.5% and 25% duty cycles and there is a pending unmasked FP exception (ES pending), if you attempt a FP load (or Intel® MMX technology Mov instruction) and the load has an longer than typical latency the processor can enter a livelock.

**Implication:** When this erratum occurs, the processor will enter a livelock condition. Intel has not observed this erratum with any commercially available software or system.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W8. Write Cycle of Write Combining Memory Type Does Not Self Snoop

**Problem:** Write cycles of WC memory type do not self-snoop. This may result in data inconsistency- if the addresses of the WC data are aliased to WB memory type memory, which has been cached. In such a case, the internal caches will not be updated with the WC data sent on the system bus.

**Implication:** This condition may result in a data inconsistency. Intel has not observed this erratum with any commercially available software, system, nor components.

**Workaround:** Software should detect via the self-snoop bit in the CPUID features flags if the processor supports a self-snooping capability. Software should perform explicit memory management/flushing for aliased memory ranges on processor that do not self-snoop.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W9. Performance Monitoring Event That Counts Floating Point Computational Exceptions (11h) Is Not Accurate

**Problem:** Performance monitoring event that counts Floating Point Compare exceptions may have inaccurate results.

**Implication:** There is no functional impact of this erratum. However this Performance Monitoring Event should not be used when accurate performance monitoring is required.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W10. Inconsistent Reporting of Data Breakpoints on FP (Intel® MMX Technology) Loads

**Problem:** The reporting of data breakpoints on either FP or MMX technology loads is dependent upon the code faulting behavior prior to the execution of the load. If there is a fault pending prior to the execution of the load and FP exceptions are enabled there is a chance that data breakpoint on successive FP/MMX technology Loads may be reported twice.

**Implication:** Software debuggers should be aware of this possibility. There should be no implications to software operated outside of a debug environment.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W11. Code Breakpoint May Be Taken after POP SS Instruction If It Is followed by an Instruction That Faults

**Problem:** A POP SS instruction should inhibit all interrupts including Code Breakpoints until after execution of the following instruction. This allows sequential execution of POP SS and MOV eSP, eBP instructions without having an invalid stack during interrupt handling. However, a Code breakpoint may be taken after POP SS if it is followed by an instruction that faults, this result in a code breakpoint being reported on an unexpected instruction boundary since both instructions should be atomic.

**Implication:** This can result in a mismatched Stack Segment and SP. Intel has not observed this erratum with any commercially available software, or system.

**Workaround:** As recommended in the *IA32 Intel® Architecture Software Developer's Manual*, the use "POP SS" in conjunction with "MOV eSP, eBP" will avoid the failure since the "Mov" will not fault.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W12. SysEnter and SysExit Instructions May Write Incorrect Requestor Privilege Level (RPL) in the FP Code Segment Selector (FCS)

**Problem:** SysEnter and SysExit instructions may write incorrect RPL in the FP Code Segment selector (FCS). As a result of this, the RPL field in FCS may be corrupted.

**Implication:** This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

**Workaround:** It is possible for BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W13. Memory Aliasing with Inconsistent A and D Bits May Cause Processor Deadlock

**Problem:** In the event that software implements memory aliasing by having two page directory entries (PDEs) point to a common page table entry (PTE) and the Accessed and Dirty bits for the two PDEs are allowed to become inconsistent, the processor may become deadlocked.

**Implication:** This erratum has not been observed with commercially available software.

**Workaround:** Software that needs to implement memory aliasing in this way should manage the consistency of the Accessed and Dirty bits.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W14. RDMSR or WRMSR to Invalid MSR Address May Not Cause GP Fault

**Problem:** The RDMSR and WRMSR instructions allow reading or writing of MSR's ( Model Specific Registers) based on the index number placed in ECX. The processor should reject access to any reserved or unimplemented MSRs by generating #GP(0). However, there are some invalid MSR addressers for which the processor will not generate #GP(0). This erratum has not been observed with commercially available software.

**Implication:** For RDMSR, undefined values will be read into EDX:EAX. For WRMSR, undefined processor behavior may result.

**Workaround:** Do not use invalid MSR addresses with RDMSR or WRMSR.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W15. FP Tag Word Corruption

**Problem:** In some rare cases, fault information generated as the result of instruction execution may be incorrect. The result is an incorrect FP stack entry.

**Implication:** This erratum may result in corruption of the FP tag word in a way that a non-valid entry in the FP stack may become valid. The software is not expected to read a non-valid entry. If the software attempts to use the stack entry (which is expected to be empty) the result may be an erroneous "Stack overflow".

**Workaround:** Do not disable SSE/SSE2 in control register CR4 and avoid code segment limit violation.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W16. Unable to Disable Reads/Writes to Performance Monitoring Related MSRs

**Problem:** The Performance Monitoring Available bit in the miscellaneous processor features MSR (IA32_MISC_ENABLES.7) was defined so that when it is cleared to a 0, RDMSR/WRMSR/RDPMC instructions would return all zeros for reads of and prevent any writes to performance monitoring related MSRs. Currently it is possible to read from or write to performance monitoring related MSRs when the Performance Monitoring Available bit is cleared to a 0.

**Implication:** It is not possible to disallow reads and writes to the Performance Monitoring MSRs. Intel has not observed this erratum with any commercially available software or system.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W17. Move to Control Register Instruction May Generate a Breakpoint Report

**Problem:** A move (MOV) to control register (CR) instruction where control register is CR0, CR3 or CR4 may generate a breakpoint report.

**Implication:** MOV to control register instruction is not expected to generate a breakpoint report.

**Workaround:** Ignore breakpoint data from MOV to CR instruction.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W18. REP MOVS Operation in Fast String Mode Continues in That Mode When Crossing into a Page with a Different Memory Type

**Problem:** A fast "REP MOVS" operation will continue to be handled in fast mode when the string operation crosses a page boundary into an Uncacheable (UC) memory type. Also if the fast string operation crosses a page boundary into a WC memory region, the processor will not self snoop the WC memory region. This may eventually result in incorrect data for the WC portion of the operation if those cache lines were previously cached as WB (through aliasing) and modified.

**Implication:** String elements should be handled by the processor at the native operand size in UC memory. In the event that the WB to WC aliasing case occurs, the end result could vary from normal software execution to potential software failure. Intel has not observed either aspects of this erratum in commercially available software.

**Workaround:** Software operating within Intel's recommendation will not require WB and WC memory aliased to the same physical address.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W19. The FXSAVE, STOS, or MOVS Instruction May Cause a Store Ordering Violation When Data Crosses a Page with a UC Memory Type

**Problem:** If the data from an FXSAVE, STOS, or MOVS instruction crosses a page boundary from WB to UC memory type and this instruction is immediately followed by a second instruction that also issues a store to memory, the final data stores from both instructions may occur in the wrong order.

**Implication:** The impact of this store ordering behavior may vary from normal software execution to potential software failure. Intel has not observed this erratum in commercially available software.

**Workaround:** FXSAVE, STOS, or MOVS data must not cross page boundary from WB to UC memory type.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W20. Machine Check Exception May Occur Due to Improper Line Eviction in the IFU

**Problem:** The processor is designed to signal an unrecoverable machine check exception (MCE) as a consistency checking mechanism. Under a complex set of circumstances involving multiple speculative branches and memory accesses there exists a one cycle long window in which the processor may signal a MCE in the instruction fetch unit (IFU) because instructions previously decoded have been evicted from the IFU. The one cycle long window is opened when an opportunistic fetch receives a partial hit on a previously executed but not as yet completed store resident in the store buffer. The resulting partial hit erroneously causes the eviction of a line from the IFU at a time when the processor is expecting the line to still be present. If the MCE for this particular IFU event is disabled, execution will continue normally.

**Implication:** While this erratum may occur on a system with any number of processors, the probability of occurrence increases with the number of processors. If this erratum does occur, a machine check exception will result. Note systems that implement an operating system that does not enable the Machine Check Architecture will be completely unaffected by this erratum (e.g., Windows* 95 and Windows*98).

**Workaround:** It is possible for BIOS code to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W21. POPF and POPFD Instructions That Set the Trap Flag Bit May Cause Unpredictable Processor Behavior

**Problem:** In some rare cases, POPF and POPFD instructions that set the Trap Flag (TF) bit in the EFLAGS register (causing the processor to enter Single-Step mode) may cause unpredictable processor behavior.

**Implication:** Single step operation is typically enabled during software debug activities, not during normal system operation.

**Workaround:** There is no workaround for single step operation in commercially available software. For debug activities on custom software the POPF and POPFD instructions could be immediately followed by a NOP instruction to facilitate correct execution.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W22. Performance Event Counter Returns Incorrect Value on L2_LINES_IN Event

**Problem:** The performance event counter returns an incorrect value on L2_LINES_IN event (EMON event #24H) when the L2 cache is disabled.

**Implication:** Due to this erratum, L2_LINES_IN performance event counter should not be monitored while the L2 cache is disabled. This erratum has no functional impact.

**Workaround:** Ignore L2_LINES_IN event when the L2 cache is disabled.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W23. VM Bit Will Be Cleared on a Double Fault Handler

**Problem:** Following a task switch to a Double Fault Handler that was initiated while the processor was in virtual-8086 (VM86) mode, the VM bit will be incorrectly cleared in EFLAGS.

**Implication:** When the OS recovers from the double fault handler, the processor will no longer be in VM86 mode

**Workaround:** None

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W24. Code Fetch Matching Disabled Debug Register May Cause Debug Exception

**Problem:** The bits L0-3 and G0-3 enable breakpoints local to a task and global to all tasks, respectively. If one of these bits is set, a breakpoint is enabled, corresponding to the addresses in the debug registers DR0-DR3. If at least one of these breakpoints is enabled, any of these registers are disabled (i.e., L*n* and G*n* are 0), and RW*n* for the disabled register is 00 (indicating a breakpoint on instruction execution), normally an instruction fetch will not cause an instruction-breakpoint fault based on a match with the address in the disabled register(s). However, if the address in a disabled register matches the address of a code fetch which also results in a page fault, an instruction-breakpoint fault will occur.

**Implication:** While debugging software, extraneous instruction-breakpoint faults may be encountered if breakpoint registers are not cleared when they are disabled. Debug software which does not implement a code breakpoint handler will fail, if this occurs. If a handler is present, the fault will be serviced. Mixing data and code may exacerbate this problem by allowing disabled data breakpoint registers to break on an instruction fetch.

**Workaround:** The debug handler should clear breakpoint registers before they become disabled.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W25. Upper Four PAT Entries Not Usable with Mode B or Mode C Paging

**Problem:** The page attribute table (PAT) contains eight entries, which must all be initialized and considered when setting up memory types for the Pentium III processor. However, in Mode B or Mode C paging, the upper four entries do not function correctly for 4-Kbyte pages. Specifically, bit 7 of page table entries that translate addresses to 4-Kbyte pages should be used as the upper bit of a 3-bit index to determine the PAT entry that specifies the memory type for the page. When Mode B (CR4.PSE = 1) and/or Mode C (CR4.PAE) are enabled, the processor forces this bit to zero when determining the memory type regardless of the value in the page table entry. The upper four entries of the PAT function correctly for 2-Mbyte and 4-Mbyte large pages (specified by bit 12 of the page directory entry for those translations).

**Implication:** Only the lower four PAT entries are useful for 4-kB translations when Mode B or C paging is used. In Mode A paging (4-Kbyte pages only), all eight entries may be used. All eight entries may be used for large pages in Mode B or C paging.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W26. SSE/SSE2 Streaming Store Resulting in a Self-Modifying Code (SMC) Event May Cause Unexpected Behavior

**Problem:** An SSE or SSE2 streaming store that results in a self-modifying code (SMC) event may cause unexpected behavior. The SMC event occurs on a full address match of code contained in L1 cache.

**Implication:** Due to this erratum, any of the following events may occur:

1. A data access break point may be incorrectly reported on the instruction pointer (IP) just before the store instruction.
2. A non-cacheable store can appear twice on the external bus (the first time it will write only 8 bytes, the second time it will write the entire 16 bytes).

Intel has not observed this erratum with any commercially available software. This erratum has been seen in a synthetic test environment.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W27. Error in Instruction Fetch Unit (IFU) Can Result in an Erroneous Machine Check-Exception (#MC)

**Problem:** A rare combination of events including the generation of a bus lock(s), the execution of a WBINVD instruction, and a page accessed or dirty bit assist may result in an erroneous Machine Check-Exception (#MC).

**Implication:** Due to this erratum, unexpected machine check-exception (#MC) is generated. Intel has not been able to reproduce this erratum with commercially available software.

**Workaround:** It is possible for BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W28. Removed; See Erratum W3.

### W29. Removed; See Erratum W4.

### W30. Removed; See Erratum W5.

### W31. Page with PAT (Page Attribute Table) Set to USWC (Uncacheable Speculative Write Combine) While Associated MTRR (Memory Type Range Register) Is UC (Uncacheable) May Consolidate to UC

**Problem:** A page whose PAT memory type is USWC while the relevant MTRR memory type is UC, the consolidated memory type may be treated as UC (rather than WC as specified in *IA-32 Intel® Architecture Software Developer's Manual*).

**Implication:** When this erratum occurs, the memory page may be as UC (rather than WC). This may have a negative performance impact.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W32. Under Certain Conditions LTR (Load Task Register) Instruction May Result in System Hang

**Problem:** A LTR instruction may result in a system hang if all the following conditions are met:

1. Invalid data selector of the TR (Task Register) resulting with either #GP (General Protection Fault) or #NP (Segment Not Present Fault).
2. GDT (Global Descriptor Table) is not 8-bytes aligned.
3. Data BP (breakpoint) is set on cache line containing the descriptor data.

**Implication:** This erratum may result in system hang if all conditions have been met. This erratum has not been observed in commercial operating systems or software. For performance reasons, GDT is typically aligned to 8-bytes.

**Workaround:** Align GDT to 8 bytes.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W33. Loading from Memory Type USWC (Uncacheable Speculative Write Combine) May Get Its Data Internally Forwarded from a Previous Pending Store

**Problem:** A load from memory type USWC may get its data internally forwarded from a pending store. As a result, the expected load may never be issued to the external bus.

**Implication:** When this erratum occurs, a USWC load request may be satisfied without being observed on the external bus. There are no known usage models where this behavior results in any negative side-effects.

**Workaround:** Do not use memory type USWC for memory that has read side-effects.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W34. FPU Operand Pointer may not be cleared following FINIT/FNINIT

**Problem:** Initializing the floating point state with either FINIT or FNINT, may not clear the x87 FPU Operand (Data) Pointer Offset and the x87 FPU Operand (Data) Pointer Selector (both fields form the FPUDataPointer). Saving the floating point environment with FSTENV, FNSTENV, or floating point state with FSAVE, FNSAVE or FXSAVE before an intervening FP instruction may save uninitialized values for the FPUDataPointer.

**Implication:** When this erratum occurs, the values for FPUDataPointer in the saved floating point image structure may appear to be random values. Executing any non-control FP instruction with memory operand will initialize the FPUDataPointer. Intel has not observed this erratum with any commercially available software.

**Workaround:** After initialization, do not expect a floating point state saved memory image to be correct, until at least one non-control FP instruction with a memory operand has been executed.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W35. FSTP (Floating Point Store) Instruction Under Certain Conditions May Result in Erroneously Setting a Valid Bit on an FP (Floating Point) Stack Register

**Problem:** When an FSTP instruction with a PDE/PTE (Page Directory Entry/Page Table Entry) A/D bit update is followed by a user mode access fault due to a code fetch to a page that has supervisor only access permission, this may result in erroneously setting a valid bit of an FP stack register. The FP top of stack pointer is unchanged.

**Implication:** This erratum may cause an unexpected stack overflow.

**Workaround:** User mode code should not depend on being able to recover from illegal accesses to memory regions protected with supervisor only access when using FP instructions.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W36. An Execute Disable Bit Violation May Occur on a Data Page-Fault

**Problem:** Under a combination of internal events, unexpected Execute Disable violations may occur on data accesses that are Execute Disable protected.

**Implication:** This erratum may cause unexpected Execute Disable violations.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W37. CPUID Leaf 0x80000006 May Provide the Incorrect Value for an 8-Way Associative Cache

**Problem:** CPUID leaf 0x80000006 may return 0x8 in ECX [15:12] to indicate 8-way associative cache, but the correct encoding for an 8-way associative cache is 0x6.

**Implication:** Software that depends on the associativity of the cache may not function correctly.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W38. Snoops during the Execution of a HLT (Halt) Instruction May Lead to Unexpected System Behavior

**Problem:** If during the execution of a HLT instruction an external snoop causes an eviction from the instruction fetch unit (IFU) instruction cache, the processor may, on exit from the HLT state, erroneously read stale data from the victim cache.

**Implication:** This erratum may lead to unexpected system behavior. Intel has only observed this condition in non-mobile configurations.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W39. Invalid Entries in Page-Directory-Pointer-Table Register (PDPTR) May Cause General Protection (#GP) Exception If the Reserved Bits Are Set to One

**Problem:** Invalid entries in the Page-Directory-Pointer-Table Register (PDPTR) that have the reserved bits set to one may cause a General Protection (#GP) exception.

**Implication:** Intel has not observed this erratum with any commercially available software.

**Workaround:** Do not set the reserved bits to one when PDPTR entries are invalid.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W40. INIT Does Not Clear Global Entries in the TLB

**Problem:** INIT may not flush a TLB entry when:

1. The processor is in protected mode with paging enabled and the page global enable flag is set (PGE bit of CR4 register)
2. G bit for the page table entry is set
3. TLB entry is present in TLB when INIT occurs

**Implication:** Software may encounter unexpected page fault or incorrect address translation due to a TLB entry erroneously left in TLB after INIT.

**Workaround:** Write to CR3, CR4 or CR0 registers before writing to memory early in BIOS code to clear all the global entries from TLB.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W41. Use of Memory Aliasing with Inconsistent Memory Type May Cause System Hang

**Problem:** Software that implements memory aliasing by having more than one linear addresses mapped to the same physical page with different cache types may cause the system to hang. This would occur if one of the addresses is non-cacheable used in code segment and the other a cacheable address. If the cacheable address finds its way in instruction cache, and non-cacheable address is fetched in IFU, the processor may invalidate the non-cacheable address from the fetch unit. Any micro-architectural event that causes instruction restart will expect this instruction to still be in fetch unit and lack of it will cause system hang.

**Implication:** This erratum has not been observed with commercially available software.

**Workaround:** Although it is possible to have a single physical page mapped by two different linear addresses with different memory types, Intel has strongly discouraged this practice as it may lead to undefined results. Software that needs to implement memory aliasing should manage the memory type consistency.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W42. Machine Check Exception May Occur When Interleaving Code between Different Memory Types

**Problem:** A small window of opportunity exists where code fetches interleaved between different memory types may cause a machine check exception. A complex set of micro-architectural boundary conditions is required to expose this window.

**Implication:** Interleaved instruction fetches between different memory types may result in a machine check exception. The system may hang if machine check exceptions are disabled. Intel has not observed the occurrence of this erratum while running commercially available applications or operating systems.

**Workaround:** Software can avoid this erratum by placing a serializing instruction between code fetches between different memory types.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W43. Split I/O Writes Adjacent to Retry of APIC End of Interrupt (EOI) Request May Cause Livelock Condition

**Problem:** When Split I/O instruction writes occur adjacent to a retry of a Local APIC End of Interrupt (EOI) request by the chipset, a livelock condition may result. The required sequences of events are:

1. The processor issues a Local APIC EOI message.
2. The chipset responds with a retry because its downstream ports are full. It expects the processor to return with the same EOI request.
3. The processor issues a Split I/O write instruction instead.
4. The chipset responds with a retry because it expected the APIC EOI.
5. The processor insists the Split I/O write instruction must be completed and issues write instruction again.

**Implication:** A processor livelock may occur causing a system hang. This issue has only been observed in synthetic lab testing conditions and has not been seen in any commercially available applications. The erratum does not occur with Intel mobile chipset based platforms.

**Workaround:** Use the PIC instead of the APIC for the interrupt controller.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W44. General Protection (#GP) Fault May Not Be Signaled on Data Segment Limit Violation above 4-G Limit

**Problem:** Memory accesses to flat data segments (base = 00000000h) that occur above the 4-G limit (0ffffffffh) may not signal a #GP fault.

**Implication:** When such memory accesses occur, the system may not issue a #GP fault.

**Workaround:** Software should ensure that memory accesses do not occur above the 4-G limit (0ffffffffh).

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W45. DR3 Address Match on MOVD/MOVQ/MOVNTQ Memory Store Instruction May Incorrectly Increment Performance Monitoring Count for Saturating SIMD Instructions Executed (Event B1h)

**Problem:** Performance monitoring for Event B1h normally increments on saturating SIMD instruction executed. Regardless of DR7 programming, if the linear address of a memory store MOVD/MOVQ/MOVNTQ instruction executed matches the address in DR3, the B1h counter may be incorrectly incremented.

**Implication:** The value observed for performance monitoring count for saturating SIMD instructions executed may be too high.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W46. Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts

**Problem:** Interrupts that are pending prior to the execution of the STI (Set Interrupt Flag) instruction are serviced immediately after the STI instruction is executed. Because of this erratum, if following STI, an instruction that triggers a #MF is executed while STPCLK#, Enhanced Intel SpeedStep® Technology transitions or Thermal Monitor 1 events occur, the pending #MF may be serviced before higher priority interrupts.

**Implication:** Software may observe #MF being serviced before higher priority interrupts.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W47. Processor INIT# Will Cause a System Hang if Triggered during an NMI Interrupt Routine Performed during Shutdown

**Problem:** During the execution of an NMI interrupt handler, if shutdown occurs followed by the INIT# signal being triggered, the processor will attempt initialization but fail soft reset.

**Implication:** Due to this erratum the system may hang.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W48. Certain Performance Monitoring Counters Related to Bus, L2 Cache and Power Management are Inaccurate

**Problem:** All Performance Monitoring Counters in the ranges 21H-3DH and 60H-7FH may have inaccurate results up to ±7.

**Implication:** There may be a small error in the affected counts.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W49. CS Limit Violation on RSM May be Serviced before Higher Priority Interrupts/Exceptions

**Problem:** When the processor encounters a CS (Code Segment) limit violation, a #GP (General Protection Exception) fault is generated after all higher priority Interrupts and exceptions are serviced. Because of this erratum, if RSM (Resume from System Management Mode) returns to execution flow where a CS limit violation occurs, the #GP fault may be serviced before a higher priority Interrupt or Exception (e.g. NMI (Non-Maskable Interrupt), Debug break(#DB), Machine Check (#MC), etc.).

**Implication:** Operating systems may observe a #GP fault being serviced before higher priority Interrupts and Exceptions.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W50. A Write to an APIC Register Sometimes May Appear to Have Not Occurred

**Problem:** With respect to the retirement of instructions, stores to the uncacheable memory-based APIC register space are handled in a non-synchronized way. For example if an instruction that masks the interrupt flag, e.g. CLI, is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt enabled flag is finally set, i.e. by STI instruction. Interrupts will remain pending and are not lost.

**Implication:** In this example the processor may allow interrupts to be accepted but may delay their service.

**Workaround:** This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### W51. The Processor May Report a #TS Instead of a #GP Fault

**Problem:** A jump to a busy TSS (Task-State Segment) may cause a #TS (invalid TSS exception) instead of a #GP fault (general protection exception).

**Implication:** Operation systems that access a busy TSS may get invalid TSS fault instead of a #GP fault. Intel has not observed this erratum with any commercially available software.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W52. Last Exception Record (LER) MSRs May be Incorrectly Updated

**Problem:** The LASTINTTOIP and LASTINTFROMIP MSRs (1DDH-1DEH) may contain incorrect values after the following events: masked SSE2 floating-point exception, StopClk, NMI and INT.

**Implication:** The value of the LER MSR may be incorrectly updated to point to a SIMD Floating-Point instruction even though no exception occurred on that instruction or to point to an instruction that was preceded by a StopClk interrupt or rarely not to be updated on Interrupts (NMI and INT).

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W53. FP Inexact-Result Exception Flag May Not Be Set

**Problem:** When the result of a floating-point operation is not exactly representable in the destination format (1/3 in binary form, for example), an inexact-result (precision) exception occurs. When this occurs, the PE bit (bit 5 of the FPU status word) is normally set by the processor. Under certain rare conditions, this bit may not be set when this rounding occurs. However, other actions taken by the processor (invoking the software exception handler if the exception is unmasked) are not affected. This erratum can only occur if the floating-point operation which causes the precision exception is immediately followed by one of the following instructions:

- FST m32real
- FST m64real
- FSTP m32real
- FSTP m64real
- FSTP m80real
- FIST m16int
- FIST m32int
- FISTP m16int
- FISTP m32int
- FISTP m64int

Note that even if this combination of instructions is encountered, there is also a dependency on the internal pipelining and execution state of both instructions in the processor.

**Implication:** Inexact-result exceptions are commonly masked or ignored by applications, as it happens frequently, and produces a rounded result acceptable to most applications. The PE bit of the FPU status word may not always be set upon receiving an inexact-result exception. Thus, if these exceptions are unmasked, a floating-point error exception handler may not recognize that a precision exception occurred. Note that this is a "sticky" bit, i.e., once set by an inexact-result condition, it remains set until cleared by software.

**Workaround:** This condition can be avoided by inserting two none floating-point instructions between the two floating-point instructions.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W54. MOV with Debug Register Causes Debug Exception

**Problem:** When in V86 mode, if a MOV instruction is executed on debug registers, a general-protection exception (#GP) should be generated, as documented in the Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide, Section 14.2. However, in the case when the general detect enable flag (GD) bit is set, the observed behavior is that a debug exception (#DB) is generated instead.

**Implication:** With debug-register protection enabled (i.e., the GD bit set), when attempting to execute a MOV on debug registers in V86 mode, a debug exception will be generated instead of the expected general-protection fault.

**Workaround:** In general, operating systems do not set the GD bit when they are in V86 mode. The GD bit is generally set and used by debuggers. The debug exception handler should check that the exception did not occur in V86 mode before continuing. If the exception did occur in V86 mode, the exception may be directed to the general-protection exception handler.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W55. Processor Digital Thermal Sensor (DTS) Readout Stops Updating upon Returning from C3 State

**Problem:** Digital Thermal Sensor (DTS) Readout is provided in IA32_THERM_STATUS bits 22:16. Upon waking up from C3 low-power state, the DTS readout will no longer be updated.

**Implication:** Upon waking up from C3 low-power state, software cannot rely on DTS readout. Any thermal threshold interrupts that are enabled in IA32_THERM_INTERRUPT, will also be affected.

**Workaround:** It is possible for BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W56. LOCK# Asserted During a Special Cycle Shutdown Transaction May Unexpectedly De-assert

**Problem:** During a processor shutdown transaction, when LOCK# is asserted and if a DEFER# is received during a snoop phase and the Locked transaction is pipelined on the front side bus (FSB), LOCK# may unexpectedly de-assert.

**Implication:** When this erratum occurs, the system may hang during shutdown. Intel has not observed this erratum with any commercially available systems or software.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W57. Last Branch Records (LBR) Updates May be Incorrect after a Task Switch

**Problem:** A Task-State Segment (TSS) task switch may incorrectly set the LBR_FROM value to the LBR_TO value.

**Implication:** The LBR_FROM will have the incorrect address of the Branch Instruction.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W58. Address Reported by Machine-Check Architecture (MCA) on Single-bit L2 ECC Errors May be Incorrect

**Problem:** When correctable single-bit ECC errors occur in the L2 cache the address is logged in the MCA address register (MCi_ADDR). Under some scenarios, the address reported may be incorrect.

**Implication:** Software should not rely on the value reported in MCi_ADDR, for Single-bit L2 ECC errors.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W59. Disabling of Single-step On Branch Operation May be Delayed following a POPFD Instruction

**Problem:** Disabling of Single-step On Branch Operation may be delayed, if the following conditions are met:

1. "Single Step On Branch Mode" is enabled (DebugCtlMSR.BTF and EFLAGS.TF are set)
2. POPFD used to clear EFLAGS.TF
3. A jump instruction (JMP, Jcc, etc.) is executed immediately after POPFD0.

**Implication:** Single-step On Branch mode may remain in effect for one instruction after the POPFD instruction disables it by clearing the EFLAGS.TF bit.

**Workaround:** There is no workaround for Single-Step operation in commercially available software. The workaround for custom software is to execute at least one instruction following POPFD before issuing a JMP instruction.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W60. Performance Monitoring Counters That Count External Bus Events May Report Incorrect Values after Processor Power State Transitions

**Problem:** Performance monitoring counters that count external bus events operate when the processor is in the Active state (C0). If a processor transitions to a new power state, these Performance monitoring counters will stop counting, even if the event being counted remains active.

**Implication:** After transitioning between processor power states, software may observe incorrect counts in Performance monitoring counters that count external bus events.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W61. VERW/VERR/LSL/LAR Instructions May Unexpectedly Update the Last Exception Record (LER) MSR

**Problem:** The LER MSR may be unexpectedly updated, if the resultant value of the Zero Flag (ZF) is zero after executing the following instructions:

1. VERR (ZF=0 indicates unsuccessful segment read verification)
2. VERW (ZF=0 indicates unsuccessful segment write verification)
3. LAR (ZF=0 indicates unsuccessful access rights load)
4. LSL (ZF=0 indicates unsuccessful segment limit load).

**Implication:** The value of the LER MSR may be inaccurate if VERW/VERR/LSL/LAR instructions are executed after the occurrence of an exception.

**Workaround:** Software exception handlers that rely on the LER MSR value should read the LER MSR before executing VERW/VERR/LSL/LAR instructions.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W62. Performance Monitoring Events for Retired Floating Point Operations (C1h) May Not Be Accurate

**Problem:** Performance monitoring events that count retired floating point operations may be too high.

**Implication:** The Performance Monitoring Event may have an inaccurate count.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W63. Global Pages in the Data Translation Look-Aside Buffer (DTLB) May Not Be Flushed by RSM instruction before Restoring the Architectural State from SMRAM

**Problem:** The Resume from System Management Mode (RSM) instruction does not flush global pages from the Data Translation Look-Aside Buffer (DTLB) prior to reloading the saved architectural state.

**Implication:** If SMM turns on paging with global paging enabled and then maps any of linear addresses of SMRAM using global pages, RSM load may load data from the wrong location.

**Workaround:** Do not use global pages in system management mode.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W64. Data Breakpoint/Single Step on MOV SS/POP SS May Be Lost after Entry into SMM

**Problem:** Data Breakpoint/Single Step exceptions are normally blocked for one instruction following MOV SS/POP SS instructions. Immediately after executing these instructions, if the processor enters SMM (System Management Mode), upon RSM (resume from SMM) operation, normal processing of Data Breakpoint/Single Step exceptions is restored.

Because of this erratum, Data Breakpoints/Single step exceptions on MOVSS/POPSS instructions may be lost under one of the following conditions.

1. Following SMM entry and after RSM, the next instruction to be executed is HLT or MWAIT
2. SMM entry after executing MOV SS/POP SS is the result of executing an I/O instruction that triggers a synchronous SMI (System Management Interrupt)

**Implication:** Data Breakpoints/Single step operation on MOV SS/POP SS instructions may be unreliable in the presence of SMIs.

**Workaround:** None Identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W65. Hardware Prefetch Performance Monitoring Events May Be Counted Inaccurately

**Problem:** Hardware prefetch activity is not accurately reflected in the hardware prefetch performance monitoring.

**Implication:** This erratum may cause inaccurate counting for all hardware prefetch performance monitoring events.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

## W66. REP MOVS Operation in Fast String Mode Continues in That Mode When Crossing into a Page with a Different Memory Type

**Problem:** When performing a "REP MOVS" instruction on WB or WC memory type the processor normally operates in the fast string move mode. Due to this erratum a “REP MOVS” instruction continues to be handled in fast mode when the string operation crosses a page boundary from a WB or WC memory type into one of the following memory types: UC, WP, and WT.

**Implication:** When in fast string mode and next page memory type is changed to:

1. UC the data size of each write is always 8 bytes, as opposed to the original data size.
2. WT there may be a memory ordering violation.
3. WP the data size of each write is always 8 bytes, as opposed to the original data size and there maybe a memory ordering violation

**Workaround:** Software should avoid crossing page boundaries from WB or WC memory type to UC,WT or WP memory type within a single "REP MOVS" instruction.

**Status:** For the steppings affected, see the Summary Tables of Changes.

§

# Specification Changes

There are no Specification Changes in this Specification Update revision.

§

# Specification Clarifications

## W1. Removed

See the Revision History for details.

## W2. Removed

See the Revision History for details.

§

# *Documentation Changes*

There are no Documentation Changes in this Specification Update revision.

§